US006418554B1

(12) **United States Patent**
Delo et al.

(10) **Patent No.: US 6,418,554 B1**
(45) **Date of Patent: Jul. 9, 2002**

(54) **SOFTWARE IMPLEMENTATION INSTALLER MECHANISM**

(75) Inventors: **John C. Delo**, Bellevue; **Malcolm S. Haar**, Seattle; **Chetan A. Parulekar**, Redmond; **Tracy D. Ferrier**, Issaquah; **Benjamin Chamberlain**, Redmond; **David E. Gonzalez**, Issaquah; **David R. Mckinnis**, Seattle, all of WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, VA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/158,021**

(22) Filed: **Sep. 21, 1998**

(51) **Int. Cl.**^{7} **G06F 9/445**
(52) **U.S. Cl.** **717/11**
(58) **Field of Search** 395/712; 717/11; 707/200, 203; 709/220

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,421,009 A	5/1995	Platt	709/221
5,473,772 A	12/1995	Halliwell et al.	717/11
5,535,326 A	7/1996	Baskey et al.	714/4

(List continued on next page.)

OTHER PUBLICATIONS

"Automating Microsoft Transaction Server Client Installation," Microsoft Corporation, URL:wysiwyg://MAIN.PRODINFO.6/http://msdn.mi . . . m/library/backgrnd/html/msdn_install.html (Jun. 1997), printed Feb. 29, 2000.

"Seagate Enterprise Management Software–East Announces Release of Desktop Management Suite; Best–of–Breed Point Solutions Integrated to Provide Heterogenous LAN Management," *Business Wire*, p. 04020056 (Apr. 2, 1996).

Dunigan, et al., *MCSE Training Guide: Windows NT Workstation 4*, New Riders Publishing, pp. 28–32, 402–405, 486–592 (Nov. 1997).

Lang, Jay., "IBM Bolsters Windows NT Reliability With Tools Suite," *Information Week*, p. A6ff (Jul. 20, 1998).

Green, "Windows Apps Need To Be Aware of Install Methods", *Network World*, p. 45 (Nov. 1994).

McKinney et al., "Win Tips Windows 9x", *Windows Magazine*, pp. 255–258 (Aug. 1998).

McNutt, "Administering X Sites", *Unix Review*, p. 45ff (Jul. 1992).

Methvin, David, "Problems? In Win98?", *Windows Magazine*, pp. 224ff (Oct. 1998).

Spanbauer, Scott, "Internet Explorer 4.0, Problem Child", *PC World*, p. 53 (Jan. 1998).

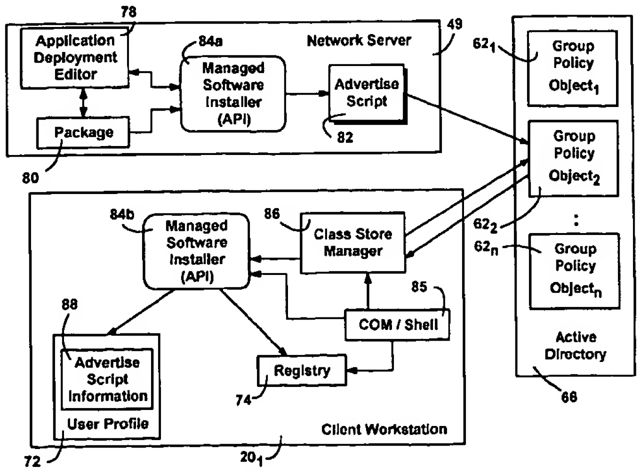
(List continued on next page.)

Primary Examiner—Kevin J. Teska
Assistant Examiner—Kelvin Booker
(74) *Attorney, Agent, or Firm*—The Law Offices of Albert S. Michalik, PLLC

(57) **ABSTRACT**

A method and mechanism for automatically installing software implementations such as applications and COM classes as they are needed from an external source. When a software implementation is needed, the mechanism first looks to the local system (e.g., registry) for that software implementation, and if found, returns the information such as a local path needed to use the software implementation. If the implementation is not found, the mechanism looks to another source, such as a CD-ROM or a centralized class store of a network, to locate the needed implementation. When located, the implementation is downloaded and locally installed from the source, and a local path is returned in a manner that is essentially transparent to the user. Software implementations such as application products may be divided into features and components to improve on-demand installation thereof.

37 Claims, 15 Drawing Sheets



U.S. PATENT DOCUMENTS

5,555,416 A	*	9/1996	Owens et al.	717/11
5,586,304 A		12/1996	Stupek, Jr. et al.	717/11
5,625,823 A	*	4/1997	Debenedictis et al.	717/6
5,630,076 A		5/1997	Saulpaugh et al.	710/104
5,644,766 A		7/1997	Coy et al.	707/204
5,655,081 A		8/1997	Bonnell et al.	709/202
5,659,547 A		8/1997	Scarr et al.	714/4
5,692,129 A	*	11/1997	Sonderegger et al.	707/103
5,732,266 A		3/1998	Moore et al.	713/1
5,732,275 A		3/1998	Kullick et al.	717/11
5,742,829 A	*	4/1998	Davis et al.	717/11
5,752,042 A		5/1998	Cole et al.	717/11
5,764,992 A		6/1998	Kullick et al.	717/11
5,768,566 A		6/1998	Harikrishnan et al.	717/11
5,778,234 A		7/1998	Hecht et al.	717/11
5,784,612 A		7/1998	Crane et al.	713/100
5,790,664 A		8/1998	Coley et al.	709/203
5,790,856 A		8/1998	Lillich	717/3
5,796,967 A		8/1998	Filepp et al.	345/339
5,805,897 A	*	9/1998	Glowny	717/11
5,835,911 A	*	11/1998	Nakagawa et al.	707/203
5,859,969 A	*	1/1999	Oki et al.	709/200
5,859,978 A	*	1/1999	Sonderegger et al.	709/226
5,867,713 A	*	2/1999	Shrader et al.	717/11
5,867,714 A	*	2/1999	Todd et al.	717/11
5,870,762 A		2/1999	Lee	707/202
5,897,640 A		4/1999	Veghte et al.	707/202
5,925,127 A		7/1999	Ahmad	713/200
5,930,513 A		7/1999	Taylor	717/11
5,930,514 A		7/1999	Thompson et al.	717/11
5,933,647 A	*	8/1999	Aronberg et al.	717/11
5,954,827 A		9/1999	Frank et al.	714/48
5,960,204 A		9/1999	Yinger et al.	717/11
5,966,540 A		10/1999	Lister et al.	717/11
5,978,590 A	*	11/1999	Imai et al.	717/11
5,987,504 A		11/1999	Toga	709/206
5,999,740 A		12/1999	Rowley	717/11
6,006,034 A		12/1999	Heath et al.	717/11
6,006,035 A		12/1999	Nabahi	717/11
6,009,274 A	*	12/1999	Fletcher et al.	717/11
6,009,401 A		12/1999	Hortsmann	705/1
6,021,438 A	*	2/2000	Duvvoori et al.	709/224
6,023,586 A		2/2000	Gaisford et al.	717/11
6,029,147 A		2/2000	Horadan et al.	705/35
6,041,333 A		3/2000	Bretschneider et al.	707/203
6,067,582 A	*	5/2000	Smith et al.	717/11
6,131,192 A		10/2000	Henry	717/11
6,151,643 A		11/2000	Cheng et al.	710/36
6,151,708 A		11/2000	Pedrizetti et al.	717/11
6,161,218 A		12/2000	Taylor	717/11
6,199,204 B1	*	3/2001	Donohue	717/11
6,202,207 B1	*	3/2001	Donohue	717/11
6,205,527 B1		3/2001	Goshey et al.	711/162
6,314,565 B1	*	11/2001	Kenner et al.	717/11

OTHER PUBLICATIONS

Anonymous, "ADSTAR Distributed Storage Manager Basic Concepts," *IBM Storage Software*, http://www.storage.ibm-.com/software/adsm/adbasics.htm pp. 1–8, (Oct. 31, 1997), printed Nov. 3, 1997.

Jones, Michael B., "The Microsoft Interactive System: An Experience Report," *Technical Report MSR–TR–97–18*, pp. 1–20, http://research.microsoft.com/~mbj/papers/mitv/tr–97–18.htm (Jul. 1997), printed Sep. 4, 2001.

* cited by examiner

reinstalled by MsiInstallMissingComponent, checking for a return code of ERROR_SUCCESS and then calling MsiLocateComponent again:

```
UINT WINAPI MsiInstallMissingComponent %(
LPCSTR% szProduct, // product code
LPCSTR% szComponent, // component ID, string GUID
INSTALLSTATE eInstallState); //local / source / default / absent invalid
```

To recover from a missing file error (i.e., an application has failed in an attempt to open a file) the general procedure of calling MsiInstallMissingFile, checking for a return code of ERROR_SUCCESS and then re-trying the file I/O operation:

```
UINT WINAPI MsiInstallMissingFile %(
LPCSTR% szProduct, // product code
LPCSTR% szFile); // filename without path
```

As can be seen from the foregoing detailed description, there is provided a method and system for automatically deploying applications across a network in accordance with a policy. Via a script associated with a policy, and applied at user logon or machine connection to the network, applications may be assigned to policy recipients (users or machines), whereby the assigned applications are advertised to those policy recipients. Other applications may be published to users, whereby the application may be indirectly activated.

While the invention is susceptible to various modifications and alternative constructions, certain illustrated embodiments thereof are shown in the drawings and have been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific form or forms disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

What is claimed is:

1. In a computer system, a method of installing a software implementation, comprising:

receiving information initiated via the computer system corresponding to a software implementation that is needed to perform a requested operation, the information comprising at least one of the set of: application extension information associated with the software implementation, application category information, an application feature identifier, and an application component identifier;

in response to receiving the information, determining that the software implementation is not installed on the computer system; and

automatically installing the software implementation on the computer system to enable the requested operation to be performed based on the determination that the software implementation is not installed on the computer system.

2. The method of claim **1** wherein determining from the information that the software implementation is not installed comprises accessing a database.

3. The method of claim **2** wherein the database comprises a system registry.

4. The method of claim **2**, wherein automatically installing the software implementation further comprises modifying the database to indicate that the software implementation is installed.

5. The method of claim **1** wherein the computer system is in a network of computers, and wherein automatically installing the software implementation comprises loading the software implementation on the computer system from a network source.

6. The method of claim **1** wherein receiving information corresponding to a software implementation comprises receiving an application identifier associated with the software implementation.

7. The method of claim **1** further comprising executing the installed software implementation and opening a file corresponding to the application extension information.

8. The method of claim **1** wherein receiving information corresponding to a software implementation comprises receiving an object identifier.

9. The method of claim **1** further comprising returning path information of the software implementation.

10. The method of claim **9** wherein the software implementation information is received from an operating system, and wherein the path is returned to the operating system.

11. The method of claim **9** wherein the software implementation information is received from an application program, and wherein the path is returned to the application program.

12. The method of claim **1** further comprising advertising a software implementation to a policy recipient.

13. The method of claim **1** wherein the computer system is connected to a network, and further comprising accessing a centralized store on the network to attempt to locate software implementation information.

14. The method of claim **1** wherein the computer system receives the information initiated via the computer system independent of a connection to a network.

15. In a computer system, a mechanism configured to provide a software implementation to a user, comprising,

an interface configured to receive information initiated via the computer system corresponding to requests to use software implementations including a software implementation that is needed to perform a requested operation but is not installed on the computer system, the interface receiving the information initiated via the computer system independent of a connection to a network;

a database describing an installed state of software implementations on the computer system; and

an installer configured to query the database in response to each requested software implementation, to identify the installed state of the requested software implementation, and to automatically install the requested software implementation if the database indicates that the requested software implementation is not installed on the computer system.

16. The mechanism of claim **15** wherein the installer comprises an install engine and an install service.

17. The mechanism of claim **16** wherein the install engine runs with user privileges and the install service runs with elevated privileges.

18. The mechanism of claim **15** wherein the computer system is in a network of computers, and further comprising a logon process, wherein the interface receives information corresponding to the installed state of the software implementation via the logon process.

19. The mechanism of claim **15** wherein the interface receives the request to use the software implementation via an operating system.

20. The mechanism of claim **15** wherein the interface receives the request to use the software implementation via an application.

21. The mechanism of claim 15 wherein the software implementation corresponds to an application program.

22. The mechanism of claim 15 wherein the software implementation corresponds to an object class.

23. The mechanism of claim 15 wherein the software implementation corresponds to an application feature.

24. The mechanism of claim 15 wherein the software implementation corresponds to an application component.

25. The mechanism of claim 24 wherein the application component comprises a dynamic link library.

26. The mechanism of claim 21 wherein the computer system is in a network of computers and wherein the installer installs the software implementation from a centralized store of information on the network.

27. A computer-readable medium having computer-executable instructions, comprising:

receiving information initiated at a computer system corresponding to a software implementation that is needed to perform a requested operation but not currently installed at the computer system;

in response to receiving the information, determining that the software implementation is available to the computer system; and

automatically installing the software implementation on the computer system to enable the requested operation to be performed.

28. The computer-readable medium of claim 27 wherein the information initiated at a computer system is received independent of a connection to a network.

29. A computer-readable medium having computer-executable instructions for performing the method of claim 1.

30. In a computer system, a method of installing a software implementation, comprising:

receiving information initiated via the computer system corresponding to a software implementation that is needed to perform a requested operation;

in response to receiving the information, determining that the software implementation is not installed on the computer system;

returning path information of the software implementation; and

automatically installing the software implementation on the computer system to enable the requested operation to be performed based on the determination that the software implementation is not installed on the computer system.

31. The method of claim 30 wherein the software implementation information is received from an operating system, and wherein the path is returned to the operating system.

32. The method of claim 30 wherein the software implementation information is received from an application program, and wherein the path is returned to the application program.

33. In a computer system in a network of computers, a mechanism configured to provide a software implementation to a user, comprising,

an interface configured to receive information initiated via the computer system corresponding to requests to use software implementations including a software implementation that is needed to perform a requested operation but is not installed on the computer system;

a database describing an installed state of software implementations on the computer system;

a logon process configured to provide information corresponding to the installed state of the software implementation to the interface; and

an installer configured to query the database in response to each requested software implementation, to identify the installed state of the requested software implementation, and to automatically install the requested software implementation if the database indicates that the requested software implementation is not installed on the computer system.

34. In a computer system, a method of installing a software implementation, comprising:

receiving, independent of a connection to a network, information initiated via the computer system corresponding to a software implementation that is needed to perform a requested operation;

in response to receiving the information, determining that the software implementation is not installed on the computer system; and

automatically installing the software implementation on the computer system to enable the requested operation to be performed based on the determination that the software implementation is not installed on the computer system.

35. In a computer system, a mechanism configured to provide a software implementation, comprising,

an interface configured to receive information initiated via the computer system corresponding to requests to use software implementations including a software implementation corresponding to an application feature that is needed to perform a requested operation but is not installed on the computer system;

a database describing an installed state of software implementations on the computer system; and

an installer configured to query the database in response to each requested software implementation, to identify the installed state of the requested software implementation, and to automatically install the requested software implementation if the database indicates that the requested software implementation is not installed on the computer system.

36. In a computer system, a mechanism configured to provide a software implementation, comprising,

an interface configured to receive information initiated via the computer system corresponding to requests to use software implementations including a software implementation corresponding to an application component that is needed to perform a requested operation but is not installed on the computer system;

a database describing an installed state of software implementations on the computer system; and

an installer configured to query the database in response to each requested software implementation, to identify the installed state of the requested software implementation, and to automatically install the requested software implementation if the database indicates that the requested software implementation is not installed on the computer system.

37. The mechanism of claim 36 wherein the application component comprises a dynamic link library.

* * * * *

US006389589B1

(12) **United States Patent**
Mishra et al.

(10) **Patent No.: US 6,389,589 B1**
(45) **Date of Patent: *May 14, 2002**

(54) **CLASS STORE SCHEMA**

(75) Inventors: **Debi P. Mishra; Markus Horstmann; Ullattil Shaji**, all of Redmond, WA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/158,023**

(22) Filed: **Sep. 21, 1998**

(51) **Int. Cl.**⁷ **G06F 9/445**; G06F 17/30

(52) **U.S. Cl.** .... **717/11**; 709/203; 709/219; 709/242; 709/244; 707/9; 707/10; 707/103 R; 707/104.1

(58) **Field of Search** .... 717/11; 713/100, 713/155, 156, 162, 182, 201, 200; 709/203, 217, 221–223, 242, 244, 239, 219; 707/10, 104, 203, 204, 9, 103 R, 103 Y, 103 Z, 103

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,421,009 A		5/1995	Platt	
5,473,772 A		12/1995	Halliwell et al.	
5,535,326 A		7/1996	Baskey et al.	
5,555,416 A		9/1996	Owens et al.	
5,586,304 A		12/1996	Stupek, Jr. et al.	717/170
5,625,823 A		4/1997	Debenedictis et al.	
5,630,076 A		5/1997	Saulpaugh et al.	
5,644,766 A	*	7/1997	Coy et al.	707/204
5,655,081 A		8/1997	Bonnell et al.	
5,659,547 A		8/1997	Scarr et al.	

(List continued on next page.)

OTHER PUBLICATIONS

Jay Lang, "IBM Bolsters Windows NT Reliability With Tools Suite—Package Provides interoperability with other platforms," Information Week, Jul. 20, 1998, pp. A6 (5 pages).*

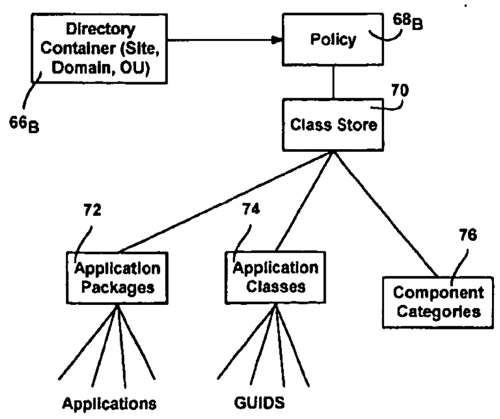
(List continued on next page.)

Primary Examiner—Tuan Q. Dam
(74) *Attorney, Agent, or Firm*—Michalik & Wylie, PLLC

(57) **ABSTRACT**

A schema that facilitates the centralized management and deployment of applications, components and services across a computer network. Centralized class stores are provided under policies associated with a directory container such as a site, domain or organizational unit. Class stores include definition, state and location information for applications and components, such that applications and components are centrally available as needed. For example, via the class store, updates to components or applications for users under an organizational unit are performed once in a centralized location, whereby users or machines may automatically obtain new versions of applications as they become available, or software implementations as needed from a centralized repository. Class stores may be configured to contain packages of component and application information according to functional areas, level of security access, or other criteria as determined by an administrator. Component categories (e.g., spreadsheet, word processor, and so on) may also be maintained, whereby a suitable application may be located by its function. For customized administration and programmatic query or installation of specific components and packages, the class store also includes a manager object that offers a set of interfaces and APIs.

63 Claims, 8 Drawing Sheets



file extension, (e.g., .XLS) and the Directory has more than one published application that can service that file type, then an administrator specified priority is used to pick the most preferred one.

In addition to applications, the COM libraries **82**, Shell **84** and Internet Explorer **86** automatically use the class store schema to install missing components. As generally represented in FIG. **7**, if a requested (step **700**) software implementation (e.g., object class) is available in the local registry (step **702**), the system provides it (step **704**). If not found, however, the system instead looks in the class stores (step **706**) for a suitable implementation and returns it if found (step **708**).

As can be seen from the foregoing detailed description, there is provided a class store schema that provides centralized information to administer a network. The schema enables the management and deployment of applications, components and services across a computer network.

While the invention is susceptible to various modifications and alternative constructions, a certain illustrated embodiment thereof is shown in the drawings and has been described above in detail. It should be understood, however, that there is no intention to limit the invention to the specific form or forms disclosed, but on the contrary, the intention is to cover all modifications, alternative constructions, and equivalents falling within the spirit and scope of the invention.

What is claimed is:

1. A computer-readable medium having stored thereon a data structure, comprising,
   (a) a first data field containing data representing an identity of an executable software module;
   (b) a second data field containing data representing network deployment state information of the executable software module in the first data field;
   (c) a third data field containing data representing network location information for the executable software module in the first data field; and
   if the network deployment state; information in the second field indicates that the executable software module identified in the first field is to be made available to a user of a client machine of the network, making the executable software module available to the user when needed, independent of which client machine the user is using, by automatically accessing the network location information and automatically installing the executable software module on the client machine based on the location information.

2. The computer-readable medium having stored thereon the data structure of claim **1** wherein the deployment information comprises state information indicative of whether the application is assigned to the policy recipient.

3. The computer-readable medium having stored thereon the data structure of claim **1** wherein the deployment state information comprises state information relating the executable software module to another executable software module.

4. The computer-readable medium having stored thereon the data structure of claim **3** wherein the deployment state information indicates that the other executable software module is to be overwritten upon installing the executable software module.

5. The computer-readable medium having stored thereon the data structure of claim **3** wherein the deployment state information indicates that the other executable software module is to be uninstalled in conjunction with installing the executable software module.

6. The computer-readable medium having stored thereon the data structure of claim **1** wherein the executable software module is advertised to the user as being available for execution on the client machine prior to installing the executable software module.

7. The computer-readable medium having stored thereon the data structure of claim **1** wherein the deployment state information comprises information indicative of whether a patch is to be installed for the executable software module.

8. The computer-readable medium having stored thereon the data structure of claim **1** wherein the deployment state information comprises information indicative of a hardware processor compatible with the executable software module.

9. The computer-readable medium having stored thereon the data structure of claim **8** wherein the hardware processor information is used to determine whether the executable software module is compatible with the client machine.

10. The computer-readable medium having stored thereon the data structure of claim **1** wherein the deployment state information comprises information indicative of an operating system compatible with the executable software module.

11. The computer-readable medium having stored thereon the data structure of claim **10** wherein the operating system information is used to determine whether the executable software module is compatible with the client machine.

12. The computer-readable medium having stored thereon the data structure of claim **1** wherein the deployment state information comprises information indicative of class identifiers implemented by the executable software module.

13. The computer-readable medium having stored thereon the data structure of claim **1** wherein the deployment state information comprises information indicative of file types handled by the executable software module.

14. The computer-readable medium having stored thereon the data structure of claim **13** wherein the information indicative of file types comprises a list of file extensions.

15. The computer-readable medium having stored thereon the data structure of claim **1** wherein the network location information comprises information indicative of a network path.

16. The computer-readable medium having stored thereon the data structure of claim **15** wherein the network path information provides a network location where application advertisement script information is present.

17. The computer-readable medium having stored thereon the data structure of claim **15** wherein the network path information provides a network location where application advertisement binaries are present.

18. The computer-readable medium having stored thereon the data structure of claim **1** wherein the deployment state information comprises information indicative of another executable software module to be upgraded by the executable software module.

19. The computer-readable medium having stored thereon the data structure of claim **1** wherein the deployment state information comprises information indicative of another executable software module that can upgrade the executable software module.

20. The computer-readable medium having stored thereon the data structure of claim **19**, wherein the deployment state information further comprises information indicative of the manner of upgrade.

21. The computer-readable medium having stored thereon the data structure of claim **20** wherein the information indicative of the manner of upgrade indicates that the upgrade will overlay the executable software module during installation of the other executable software module.

22. The computer-readable medium having stored thereon the data structure of claim 20 wherein the information indicative of the manner of upgrade indicates that the upgrade will uninstall the executable software module before installation of the other executable software module.

23. The computer-readable medium having stored thereon the data structure of claim 22 wherein the deployment state information comprises information indicative of a time when the executable software module was deployed.

24. The computer-readable medium having stored thereon the data structure of claim 1 wherein the deployment state information comprises information indicative of a time when the executable software module had its deployment state changed.

25. The computer-readable medium having stored thereon the data structure of claim 1 wherein the deployment state information comprises information indicative of whether the executable software module is to be orphaned on the user's machine.

26. The computer-readable medium having stored thereon the data structure of claim 1 wherein the deployment state information comprises information indicative of a uniform resource locator providing a location where further information can be found about this executable software module.

27. A computer-readable medium having stored thereon a data structure, comprising:

a package capable of being deployed to a policy recipient;

an object class container including registration information corresponding to the package:

component categories information of the object class, including deployment state information corresponding to the package and network location information corresponding to a location of the package, and

if the deployment state information indicates that the package is to be made available to the policy recipient, deploying the package to the policy recipient when needed by automatically accessing the network location information and automatically installing the package from the network to the policy recipient machine based on the network location information.

28. The data structure of claim 27 wherein the package includes an application program.

29. The data structure of claim 27 wherein the package includes a set of binary component implementations.

30. The data structure of claim 27 wherein the package includes a standalone component.

31. The data structure of claim 27 wherein the package includes version information.

32. The data structure of claim 27 wherein the deployment state information includes data indicating that the package is published to at least one policy recipient.

33. The data structure of claim 27 wherein the deployment state information includes data indicating that the package is assigned to at least one policy recipient.

34. In a computer network having a plurality of client machines and a plurality of policy recipients comprising client users or client machines, a method comprising:

associating policy data with a policy recipient, the policy data including at least one class store containing application deployment information corresponding to the policy recipient, the application deployment information including an identity of at least one executable software module and a state of deployment therefor; and

managing which executable software is available for use on a particular client machine based on the policy data by:

1) accessing the at least one class store to determine the state of deployment information for a selected executable software module identified therein;

2) comparing an actual state of deployment of the selected executable software module on the client machine to the state of deployment information for the selected executable software module determined from the at least one class store; and

3) automatically configuring the client machine so that the actual state of deployment of the selected executable software module corresponds to the state of deployment information in the at least one class store.

35. The method of claim 34, further comprising, evaluating access control information associated with the at least class store.

36. The method of claim 34, wherein the policy recipient is a user, and wherein the policy data associated with that user makes at least one application program available to the user independent of which client machine in the network the user is using.

37. The method of claim 34, wherein automatically configuring the client machine occurs for the user each time the user logs onto that client machine.

38. The method of claim 34, wherein the policy recipient is a client machine, and wherein the policy data associated with that client machine makes at least one application program available to any user of the client machine.

39. The method of claim 34, wherein accessing the at least one class store indicates that the selected executable software module should not be installed on the client machine, wherein comparing an actual state of deployment indicates that the selected executable software module is installed, and wherein configuring the client machine includes uninstalling the selected executable software module from the client machine.

40. The method of claim 34, wherein configuring the client machine includes writing data related to the selected executable software module to a database maintained at the client machine.

41. The method of claim 40, wherein writing data to the database includes modifying a registry on the client machine.

42. The method of claim 34, wherein the selected executable software module comprises an application program, wherein configuring the client machine includes writing data related to the application program to the client machine to advertise the application as available for execution on the client machine regardless of whether it is actually installed on the client machine, and, if the application is later requested to be executed but is not installed for execution, automatically installing and executing the application on the client machine.

43. The method of claim 34, wherein configuring the client machine includes publishing the selected executable software module to the client machine by maintaining information on the client machine for automatically installing the selected executable software module on the client machine and executing it if later needed.

44. The method of claim 34, wherein accessing the at least one class store indicates that the selected executable software module should be patched on the client machine, and wherein configuring the client machine includes patching the selected executable software module on the client machine.

45. The method of claim 34, wherein accessing the at least one class store indicates that the selected executable soft-

ware module should be overwritten on the client machine with another version thereof, and wherein configuring the client machine includes overwriting the selected executable software module on the client machine with the other version.

46. The method of claim **34**, wherein accessing the at least one class store indicates that the selected executable software module should be uninstalled and replaced with another version thereof on the client machine, and wherein configuring the client machine includes uninstalling the selected executable software module and thereafter installing the other version of the selected executable software module on the client machine.

47. The method of claim **34**, further comprising, receiving data corresponding to a class identifier at the client machine, querying the at least one class store to attempt to find executable software module deployment information associated with the class identifier, and if the class store includes deployment information for that class identifier, installing the executable software module based on the deployment information for that class identifier.

48. The method of claim **34** further comprising, receiving data corresponding to a file type at the client machine, querying the at least one class store to attempt to find executable software module deployment information associated with the file type, and if the class store includes deployment information for that file type:

a) installing the executable software module based on the deployment information for that file type in the class store, and

b) executing the executable software module.

49. The method of claim **34**, wherein the deployment information in the class store further includes installation information of the selected executable software module, wherein accessing the at least one class store indicates that the selected executable software module should be installed on the computer system, wherein comparing the actual state of deployment indicates that the selected executable software module is not installed, and wherein configuring the computer system includes installing the selected executable software module on the computer system based on the installation information.

50. A computer-readable medium having computer-executable instructions for performing the method of claim **34**.

51. The method of claim **34**, wherein the selected executable software module comprises an application program, and wherein configuring the computer system includes, advertising the application by making the application appear available for execution on the computer system without installing the selected executable software module onto the computer system until needed.

52. In a computer network, a method comprising:

associating a class store with a policy recipient, the class store containing an identity of an executable software module and network location information of the executable software module;

receiving a request, at a client machine having a relationship with the policy recipient, to execute the executable software module;

querying for information, in a system to registry of the client machine, for executing the executable software module, and if the information needed to execute the executable software module is not found:

1) querying the class store for information corresponding to the executable software module;

2) installing the executable software module based on the network location information in the class store; and

3) executing the executable software module.

53. The method of claim **52** wherein the policy recipient corresponds to the client machine.

54. The method of claim **52** wherein the policy recipient corresponds to a user logged onto the client machine.

55. The method of claim **52** wherein receiving a request includes receiving a class identifier.

56. The method of claim **52** wherein receiving a request includes receiving file type data.

57. The method of claim **52** wherein installing the executable software module based on the network location information in the class store comprises, downloading at least one file from a network source.

58. In a computer network having a plurality of client machines and a plurality of policy recipients comprising client users or client machines, a system comprising:

a policy object, the policy object containing a class store, the class store containing application deployment information including an identity of an executable software module a state of deployment for the executable software module, and network location information for the executable software module;

a selected client machine having a relationship with a policy recipient that is associated with the policy object, the client machine having access to the policy object; and

executable code on the client machine that receives a request corresponding to execution of the executable software module, wherein based on the request, the executable code:

1) accesses the class store to obtain the state of deployment information for the executable software module;

2) determines from the state of deployment information in the class store whether the policy recipient is able to execute the executable software module, and if so, automatically installs and executes the executable software module based on the network location information in the class store.

59. The system of claim **58** wherein the selected client machine corresponds to the policy recipient and the policy object comprises a machine policy object.

60. The system of claim **58** wherein a user logged onto the selected client machine corresponds to the policy recipient.

61. The system of claim **58** herein the executable code on the client machine comprises operating system code.

62. The system of claim **58** wherein the request corresponding to execution of the executable software module includes class identifier data.

63. The system of claim **58** wherein the request corresponding to execution of the executable software module includes file type data.

* * * * *